```
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Module: Math
;>
;>      BANK 1 MODULE
;>
;>      This module contains all the special purpose math routines
;>      used in the Widget firmware.
;>
;>      FUNCTION Div3_19( A : 3 BYTES { !rC:E } ) : 3 BYTES { !r0:2 }
;>      FUNCTION Div3_38( A : 3 BYTES { !rC:E } ) : 3 BYTES { !r0:2 }
;>      FUNCTION Div3_76( A : 3 BYTES { !rC:E } ) : 3 BYTES { !r0:2 }
;>      FUNCTION MulR0_m( R0 : BYTE { !r0 } ) : 3 BYTES { !r0:2 }
;>      FUNCTION DivHdsSctrs( A : 3 BYTES { !r0:2 } ) :
;>                      Quotient : BYTE { !r0:1 }
;>                      Remainder : BYTE { !r3 }
;>      FUNCTION DivSctrs( A : BYTE { !r3 } ) :
;>                      Quotient : BYTE { !r2 }
;>                      Remainder : BYTE { !r3 }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: Div3_k, Div3_19, Div3_38, Div3_76
;>
;>      This function performs the following 24 bit arithmetic operations:
;>
;>      Div3_19 : Result <-- A DIV 19
;>
;>      Div3_76 : Result <-- A DIV 76
;>
;>      Inputs:
;>              A: 3 BYTES { !rC, !rD, !rE }
;>
;>      Outputs:
;>              Result: 3 BYTES { !r0, !r1, !r2 }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN
;>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Div3_19: Quotient := Dividend DIV 19
;>
;>      Local Variables:
;>              Divisor : 3 BYTES { !rC:E }
;>              Quotient : 3 BYTES { !r7:9 }
;>              Dividend : 3 BYTES { !r4:6 }
;>
;>      BEGIN
;>       Divisor := $013000 { 19 * 2^( 13 - 1 ) }
```

```
;>       Qoutient := 0
;>       FOR i := 1 TO 13 DO
;>        IF ( Dividend >= Divisor )
;>         THEN
;>              Dividend := Dividend - Divisor
;>              Quotient := ( Quotient * 2 ) + 1
;>         ELSE Quotient := ( Quotient * 2 )
;>       Divisor := Divisor DIV 2
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>
Div3_76:        Ld      ScrRegC,#$04 ;Divisor := $04C000
                Ld      ScrRegD,#$C0
                Ld      ScrRegE,#$00
                Jr      Div3_x

Div3_38:        Ld      ScrRegC,#$02 ;Divisor := $026000
                Ld      ScrRegD,#$60
                Ld      ScrRegE,#$00
                Jr      Div3_x


Div3_19:        Ld      ScrRegC,#$01 ;Divisor := $013000
                Ld      ScrRegD,#$30
                Ld      ScrRegE,#$00

Div3_x:         Ld      ScrReg4,!rC ;pass dividend to routine
                Ld      ScrReg5,!rD
                Ld      ScrReg6,!rE

                Push    Rp ;save context
                Srp     #Wrk_Scr

                Clr     !r7 ;Quotient := 0
                Clr     !r8
                Clr     !r9

                Ld      !rA,#13 ; FOR i := 1 TO 13 DO

Div3_19_Lp:     Rcf             ;clear carry
                Rlc     !r9 ;Quotient := Quotient * 2
                Rlc     !r8
                Rlc     !r7

                 Ld     !r0,!r4 ;IF ( Dividend >= Divisor )
                 Ld     !r1,!r5
                 Ld     !r2,!r6
                Call    Sub3
                Jr      C,Div3_19_Div2

                Ld      !r4,!r0 ;Dividend := Dividend - Divisor
                Ld      !r5,!r1
                Ld      !r6,!r2

                Add     !r9,#1 ;quotient := quotient + 1
                Adc     !r8,#0
                Adc     !r7,#0

Div_3_19_Div2:  Rcf             ;clear carry flag
                Rrc     !rC ;Divisor := Divisor DIV 2
                Rrc     !rD
                Rrc     !rE
```

```
                Djnz    !rA,Div3_19_Lp

                Pop     Rp ;context switch
                Ld      !r0,ScrReg7 ;pass quotient to caller
                Ld      !r1,ScrReg8
                Ld      !r2,ScrReg9

                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: MulR0_m  { Multiply R0 by m }
;>
;>      This function muliplies the byte in !r0 by m where:
;>
;>      For 10Mbyte Profile: m = 256
;>      For 20MByte Profile: m = 512
;>      For 30MByte Profile: m = 1024
;>
;>      Inputs:
;>              A : BYTE { !r0 }
;>
;>      Outputs:
;>              Result : 3 BYTES { !rC, !rD, !rE }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .FIN
                .DO     Internal
                .LSTON
                .FIN

MulR0_m:
                Clr     !rE     ;Result := !r0 * 256
                Ld      !rD,!r0
                Clr     !rC

                .LSTOFF
                .DO     W_20MB + W_40MB
                .LSTON

                Rlc     !rE     ;Result := Result * 2
                Rlc     !rD
                Rlc     !rC

                .LSTOFF
                .FIN
                .DO     W_40MB
                .LSTON
                Rlc     !rE     ;Result := Result * 2
                Rlc     !rD
                Rlc     !rC

                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .FIN

                Jp      Bank_Ret
```

```
                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: DivHdsSctrs  { Divide by Heads * Sectors }
;>
;>      This function takes the 24 bit word passed to it and returns
;>      a 2 byte integer ( representing the cylinder number ) and
;>      the remainder from the divide.
;>
;>      Inputs:
;>              A : 3 BYTES { !rC, !rD, !rE }
;>
;>      Outputs:
;>              Cylinder : WORD { !!r0 }
;>              Remainder : 1 BYTE { !r3 }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .FIN

DivHdsSctrs:
                .DO     W_10MB
                Call    Div3_38         ;return with !r0:2 = result
                                        ;            ScrReg4:6 = remainder
                .FIN
                .DO     W_20MB + W_40MB
                Call    Div3_76         ;return with !r0:2 = result
                                        ;            ScrReg4:6 = remainder
                .FIN

                Ld      !r0,!r1 ;HiCyl := ..
                Ld      !r1,!r2 ;LoCyl := ..
                Ld      !r3,ScrReg6 ;Remainder := ..
                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: DivSctrs  { Divide by Sectors }
;>
;>      This function takes the 1 byte word passed to it and returns
;>      a 1 byte integer ( representing the head number ) and
;>      the remainder from the divide ( the remainder is the sector # )..
;>
;>      Inputs:
;>              A : 1 BYTE { !r3 }
;>
;>      Outputs:
;>              Quotient : 1 BYTE { !r2 }
;>              Remainder : 1 BYTE { !r3 }
;>
;>      Algorithm:
;>
```

```
;>              Note that there are only 2 heads on the widget, therefore
;>              if A > NumberOfSectors then Head := 1, else Head := 0
;>          and if A > NumberOfSectors then Sector := A - NumberOfSectors
;>                      else Sector := A
;>              Also, the sector is logically remapped to account for
;>              different inteaved drives. The mapping table is located
;>              in the SpareTable: when the physical sector is found it's
;>              value is used as an index into the Map-Table and the value
;>              stored at that location of the table becomes the new sector.
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

DivSctrs:
                Ld      !r0,#NbrSctrs   ;load number of sectors into !r0

                Cp      !r3,!r0         ;check for A greater/equal
                Jr      Lt,D_Sctrs1

                Ld      !r2,#1          ;select head 1
                Sub     !r3,!r0
                Jr      D_Sctrs2

D_Sctrs1:       Clr     !r2             ;select head 0

D_Sctrs2:       Ret


ReMap_Sector:   Push    !rE ;save regs
                Push    !rF
                Ld      !rE,#.HIBYTE. Map_Table
                Ld      !rF,#.LOWBYTE. Map_Table
                Add     !rF,!r0 ;index into table
                Adc     !rE,#0
                Lde     !r0,@!!rE ;get remapped sector
                Pop     !rF
                Pop     !rE

                Jp      Bank_Ret

                .LSTOFF
```